About Code Layout for Lisp

Michael L. Gassanenko
http://www.forth.org.ru/~mlg

1 Introduction

Layout is also a language, the language of structure.

2 Problem

Lots of Idiotic, Silly (closing) Parentheses [1] do not reflect the structure of the program.

3 Solution

Place the closing parentheses on a separate line, place each closing paren below an opening paren.

```
; in Scheme, from
; http://dec.bournemouth.ac.uk/forth/euro/ef99/gassanenko99a.pdf
(define free
 (lambda (mem addr) ; --> mem'
    (define remove-assoc
     (lambda (newmem oldmem addr)
      (cond
        ( (null? oldmem) newmem )
        ( (equal? (caar oldmem) addr)
              (remove-assoc newmem (cdr oldmem) addr)
        )
        ( else
              (remove-assoc
                  (cons (car oldmem) newmem)
                  (cdr oldmem)
                  addr
    ))) )     )
    (remove-assoc () mem addr)
))
```

More precisely, we arrange closing parens so that the reader can find the end of the block started by each opening paren by either examining the column below the opening paren, or, if and only if there is no corresponding paren in the column, by examining the current line.

Note that the closing parens are in reverse order. The notation

```
(a (b (c (d
) ) ) )
```

[1] some people sometimes say that this is what LISP stands for

has in actuality the structure of

```
(a [b {c <d
> } ] )
```

Two facts make our notation possible:

- the closing parens are identical;
- when we read code, what we want to know is not correspondence between opening and closing parens, but which structures are closed and which remain open.

There is no need to move all closing parens onto separate paren lines. The intent of this layout is to have block boundaries evident; if this goal is already reached, further reformatting in this manner cannot improve readability. What must be avoided is such arrangement of parens that the other block boundary cannot be found by either vertical or horizontal movement of the eye alone.

For example, when looking at

```
(cons (car oldmem) newmem)
(cdr oldmem)
```

the reader sees that there is no closing paren under the first opening paren, so the corresponding paren must be on the same line.

In the case of

```
( (equal? (caar oldmem) addr)
      (remove-assoc newmem (cdr oldmem) addr)
)
```

the first opening paren corresponds to the closing paren below it; the second opening paren has no closing paren below it so the closing paren must be at the same line. In this example we had to place the last closing paren on a separate line because the corresponding opening paren is on a different line.

4 Possible Confusion

The code is easy to read, but modification requires attention: in the following example

```
(a
 (b
   (c
       (d
         ...
)) )   )
```

we cannot add z to the argument list of c by inserting z as below:

```
(a
 (b
   (c
       (d
         ...
)) )z  ) ; misleading: z is passed to a
```

because the first closing parenthesis corresponds to the last opening parenthesis.
Instead, we have to split the line first:

```
(a
 (b
   (c
       (d
         ...
       )
)) )
```

and only then insert the additional argument:

```
(a
 (b
   (c
     (d
        ...
     )
     z
)) )
```

It may be recommended to define an editor macro that splits the line at cursor as shown above, and maybe a macro for joining lines.

5 Emacs Macros

```
;; reverse split line at cursor position
(fset 'reverse-split-line
   [?\C-  ?\C-a ?\M-w ?\M-x ?c ?l ?e ?a ?r ?- ?r ?e ?c ?t ?a ?n ?g ?l ?e return ?\C-e return ?\C-y]
)
;; reverse join two lines (the cursor is at the 2nd line)
(fset 'reverse-join-lines
   [?\C-  ?\C-p ?\C-a ?\M-x ?u ?n ?t ?a ?b ?i ?f ?y return ?\C-x
    ?\C-x ?\C-p ?\C-k ?\C-n ?\C-y ?\M-^ ?\C-x ?\C-x]
)
```

6 FAR Manager Macros

Line split (cursor is at the split position): Shift-End Ctrl-X Home Return End CursorUp Shift-Ins Ctrl-U CursorDown End (I assign this sequence to Alt-2)
Line join (cursor is at the end of 2nd line): CursorUp Shift-Home Ctrl-X Shift-End Ctrl-X Del End Shift-Ins Ctrl-U (I assign this sequence to Alt-1)

7 Conclusion

The rule of thumb is:

- place the closing parens on a separate line, each one below an opening paren;
- the line with closing parens must contain only closing parens;

- a small amount of closing parens may occur on a line if they all correspond to opening parens from the same line.

Not all closing parens need be moved to a separate line: the reader assumes that if there is no matching closing paren below an opening paren, the corresponding closing paren may be found on the same line.

When you modify code, mind that the parens are in actual fact in the reversal order. Define an editor macro to split the parens line.